CS 221: Computational Complexity **Prof. Salil Vadhan**

Lecture Notes 14

March 22, 2010 **Scribe: Kyu Bok Lee**

1 Agenda

- #**P**-completeness (cont.)
- Toda's Theorem
- Approximate Counting vs. Uniform Sampling (more on next lecture)

2 #P-Completeness (cont.)

Definition 1 *For an $n \times n$ matrix M, the* permanent *of M is defined to be*

$$\mathrm{perm}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} M_{i\sigma(i)}$$

where S_n is the group of permutations from $[n]$ to $[n]$.

Definition 2 *In a graph $G = (V, E)$, a* matching *is a subset of edges $E' \subseteq E$ such that each vertex in V is incident to at most one edge in E'. A* perfect matching *is when each vertex is incident to exactly one edge in E'.*

Theorem 3 (Valiant) *Computing the permanent of $\{0,1\}$-matrices is #**P**-complete.*

Proof: First we show that the problem is in #**P**. Note

$$\{0,1\}\text{-matrix} \longleftrightarrow \text{bipartite graph with } n \text{ vertices on each side,}$$
$$\text{nonzero term in perm}(M) \longleftrightarrow \text{perfect matching in } G.$$

So we have $\mathrm{perm}(M) =$ the number of perfect of matchings in G. (More generally, if M is not necessarily a $\{0,1\}$ matrix, then we can think of M as describing a *weighted* bipartite graph, and then $\mathrm{perm}(M)$ is a weighted sum of perfect matchings, where the weight of a matching is the product of the edge weights in it.) And the problem of counting perfect matchings in an arbitrary bipartite graph G is clearly in #**P**.

We now proceed to show it is #**P**-hard:

1. #SAT $\leq$ permanent of integer matrices
2. integer matrices $\leq$ nonnegative integer matrices

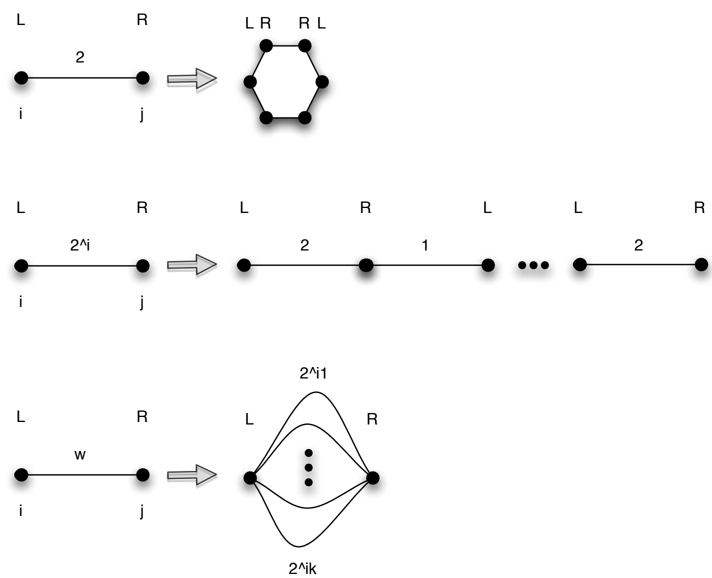
3. nonnegative integer matrices $\leq \{0,1\}$-matrices

We discuss each step below:

1. Step 1 uses fancy gadgetry, and we will not discuss details here. (See Arora–Barak if you are interested.)

2. Given $M \in \mathbb{Z}^{n\times n}$, let $v = \max_{i,j} |M_{ij}|$ and define $Q := 2v^n n! > 2 \cdot |\mathrm{perm}(M)|$. Then $\mathrm{perm}(M)$ can be computed from $\mathrm{perm}(M)$ mod Q. We can replace every negative entry M_{ij} with $Q + M_{ij}$. This nonnegative matrix M' has the property that $\mathrm{perm}(M')$ mod $Q = \mathrm{perm}(M)$ mod Q.

3. We replace weighted edges with unweighted ones as follows:



where $w = 2^{i1} + \cdots + 2^{ik}$.

The L's and R's indicate which vertices are on the left or right side of the bipartite graph. The first gadget is equivalent to a single edge of weight two, because there are two ways to match all the vertices of the gadget (including the original vertices i and j), but only one way to match the four "internal" vertices of the gadget without matching the "external" vertices (i, j). Thus, each matching in the graph that uses edge i, j gets mapped to two matchings in the new graph, and each matching in the graph that doesn't use i, j gets mapped to one matching in the new graph. ■

3 Toda's Theorem

Theorem 4 (Toda) $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$.

Proof Outline::

1. $\mathbf{PH} \leq_r \bigoplus\mathbf{P}$ (randomized karp reduction with exponentially small error)

   $\bigoplus\mathbf{P}$ is a problem of deciding whether or not there are an even or odd number of witnesses.

2. If $L \leq_r \bigoplus\mathbf{P}$ with exponentially small error, then $L \in \mathbf{P}^{\#\mathbf{P}}$. Intuitively, counting is more powerful than both computing parities and randomization.

■

Remark 5 $\mathbf{NP} \leq_r \bigoplus\mathbf{P}$ by Valiant-Vazirani Theorem.

#P vs. PP

Definition 6 $L \in \mathbf{PP}$ *if there exists a polynomial-time machine M and a polynomial p such that*

$$x \in L \iff \left|\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\}\right| > \frac{2^{p(|x|)}}{2}.$$

Proposition 7 $\mathbf{P}^{\mathbf{PP}} = \mathbf{P}^{\#\mathbf{P}}$.

Proof: One direction $\subseteq$ is easy since we can count exactly in $\mathbf{P}^{\#\mathbf{P}}$. It remains to show $\mathbf{P}^f \in \mathbf{P}^{\mathbf{PP}}$ for any $f \in \#\mathbf{P}$. Let M be the verifier and p a polynomial associated with f.

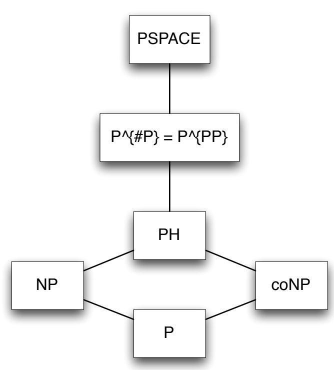
1. Define $L := \{(x,t) : \left|\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\}\right| > t\}$. This language is in $\mathbf{PP}$ since we can let our polynomial-time machine in the definition be

$$M'((x,t),(y,b)) = \begin{cases} 1 & b = 0 \text{ and } M(x,y) = 1 \text{ or} \\ & b = 1 \text{ and } y \leq \frac{2^{p(|x|)+1}}{2} - t \\ 0 & \text{otherwise} \end{cases}$$

2. Now $\mathbf{P}^f \subseteq \mathbf{P}^L$ by binary search.

■

So now we have:



4 Approximate Counting

Definition 8 *Let* $f : \{0,1\}^* \rightarrow \mathbb{N}$ *and* $\alpha \geq 1$. *An* α-approximation algorithm *for* f *is an algorithm* A *such that*

$$f(x) \leq A(x) \leq \alpha \cdot f(x)$$

for all x.

Definition 9 *If* A *is a probabilistic algorithm such that*

$$Pr[f(x) \leq A(x) \leq \alpha \cdot f(x)] \geq 2/3$$

for all x*, then we call* A *a* randomized α-approximation algorithm.

Definition 10 *An* approximation scheme *for* f *is a set of* $(1+\varepsilon)$*-approximation algorithms for all* $\varepsilon > 0$.

Definition 11 *A* fully polynomial approximation scheme *for* f *is a set of* $(1+\varepsilon)$*-approximation algorithms* $A_\varepsilon(x)$ *running in time* $\text{poly}(|x|, 1/\varepsilon)$ *for all* $x \in \{0,1\}^*$ *and all* $\varepsilon > 0$.

It turns out that there are many #**P**-complete functions with fully polynomial approximation schemes. Thus, even though exact counting is usually hard, approximate counting is often much easier.